

# Agent-based Fraud and Intrusion Detection in Financial Information Systems \*

Salvatore J. Stolfo  
David W. Fan  
Andreas Prodromidis<sup>†</sup>  
Wenke Lee  
and

Shelley Tselepis  
Department of Computer Science  
Columbia University  
New York, NY 10027  
212-939-7080

<sal,wfan,wenke,andreas,sat@cs.columbia.edu>

Philip K. Chan  
Computer Science  
Florida Institute of Technology  
Melbourne, FL 32901  
<pkc@cs.fit.edu>

November 24, 1997

---

\* This research is supported by the Intrusion Detection Program (BAA9603) from DARPA (F30602-96-1-0311), NSF (IRI-96-32225 and CDA-96-25374) and NYSSTF (423115-445).

<sup>†</sup>Supported in part by IBM

## Abstract

A secured and trusted inter-banking network for electronic commerce requires high speed verification and authentication mechanisms that allow legitimate users easy access to conduct their business, while thwarting fraudulent transaction attempts by others. Fraudulent electronic transactions are already a significant problem, one that will grow in importance as the number of access points in the nation's financial information system grows, especially when transactions are fully enabled on the internet for electronic commerce. We propose a new approach to protect the nation's financial systems from threats. This new method of protection consists of *pattern-directed inference systems* using models of anomalous or errant transaction behaviors to forewarn of impending threats. This paper describes our preliminary results using credit card fraud data supplied by our collaborators, a consortium of banks. This data provides an excellent snap shot of real-world transaction data from a context that is likely close to what will ultimately be enabled on the internet. A general purpose distributed data mining architecture is currently under development at Columbia University called *JAM* (*Java Agents for Meta-Learning*). We report a series of experiments using JAM, and its component learning programs, were performed on the credit card fraud data to guide us in architecting the best possible distributed fraud detection system.

## 1 Introduction

A secured and trusted inter-banking network for electronic commerce requires high speed verification and authentication mechanisms that allow legitimate users easy access to conduct their business, while thwarting fraudulent transaction attempts by others. Fraudulent electronic transactions are already a significant problem, one that will grow in importance as the number of access points in the nation's financial information system grows, especially when transactions are fully enabled on the internet for electronic commerce.

Financial institutions today typically develop custom fraud detection systems targeted to their own asset bases. Recently though, banks have come to realize that a unified, global approach is required, involving the periodic sharing with each other of information about attacks. Only in this way is there hope for maintaining the long-term integrity of the global financial network. Global commerce is subject to severe disruption by an enemy - either domestic or foreign - who penetrates the financial transaction system, or partially disables it. Periodic sharing of information cannot protect against such a problem. However, a widely deployed system that detects local fraudulent transaction attempts and propagates the attack information can be a highly effective countermeasure. A global (as opposed to local) approach can effectively differentiate between isolated fraud attempts and a concerted attack on the financial transaction network and react appropriately. Such an approach could be particularly useful if the detection system can be relied upon to automatically shut out the attacker.

One part of such an approach would be protecting the routers and network infrastructure in a financial information system. However intrusions and fraudulent transactions will inevitably leak through such protection to attack assets. We propose another wall to protect the nation's financial systems from threats. This

new wall of protection consists of *pattern-directed inference systems* using models of anomalous or errant transaction behaviors to forewarn of impending threats. This approach requires analysis of large and inherently distributed databases of information about transaction behaviors to produce models of “probably fraudulent” transactions. It is an emerging technique, people in AT&T, NYNEX and GTE are using this method to develop cellular phone fraud detecting system on centralized databases [18].

The key difficulties in this approach are: financial companies don’t share their data for a number of (competitive and legal) reasons; the databases that companies maintain on transaction behavior are huge and growing rapidly; real-time analysis is highly desirable to update models when new events are detected and easy distribution of models in a networked environment is essential to maintain up to date detection capability.

We propose a novel system to address these difficulties and thereby protect against electronic fraud. Our system has two key component technologies: *local fraud detection agents* that learn how to detect fraud and provide intrusion detection services within a single corporate information system, and a secure, integrated *meta-learning system* that combines the collective knowledge acquired by individual local agents.

Once derived *local classifier agents* or *models* are produced at some site(s), two or more such agents may be composed into a new classifier agent by a *meta-learning agent*. Meta-learning is a general strategy that provides the means of learning how to combine and integrate a number of separately learned classifiers or models, (each of which in the current context is a remote agent). The meta-learning system proposed will allow financial institutions to share their models of fraudulent transactions by exchanging classifier agents in a secured agent infrastructure. But they will not need to disclose their proprietary data. In this way their competitive and legal restrictions can be met, but they can still share information. The meta-learned system can be globally constructed, or alternatively it can be local. In the latter guise, each corporate entity benefits from the collective knowledge by using its privately available data to locally learn a meta-classifier agent from the shared models. In either approach, the meta-classifiers then act as sentries forewarning of possibly fraudulent transactions and threats by inspecting, classifying and labeling each incoming transaction.

This paper describes our preliminary results using credit card fraud data supplied by our collaborators, a consortium of banks. This data provides an excellent snap shot of real-world transaction data from a context that is likely close to what will ultimately be enabled on the internet. A general purpose distributed data mining architecture is currently under development at Columbia University called *JAM* (*Java Agents for Meta-Learning*). See <http://www.cs.columbia.edu/~sal/JAM/PROJECT> for a complete specification of the system, as well as its source code. A series of experiments using JAM, and its component learning programs, were performed on the credit card fraud data to guide us in architecting the best possible distributed fraud detection system. The results are reported below, followed by a series of additional questions that remain open and issues that need to be explored that are the subject of our current experimental work. For the present paper, we concern ourselves with the accuracy of the models or classifiers computed by JAM for the credit card fraud data.

## 2 Fraud Detection

### 2.1 Goal

The primary demonstrable task that we propose to attack with this new technology is **intrusion and fraud detection in financial information systems**.

The Financial Services Technology Consortium (FSTC) is a 100 member consortium of some of the nation's largest banks and information technology vendors (browse <http://www.fstc.org>). The FSTC seeks to attack the problem of fraudulent electronic transactions from a unified, global viewpoint. By contrast, the current evolving state of practice is for individual financial institutions to develop custom fraud detection systems targeted at fraudulent transactions within their own data flows.

Despite the need for privacy and the drive to protect competitive market positions, financial institutions have a strong incentive to make a concerted, cross-institution, attack on fraud. The electronic fraud problem is already significant, and it is perceived as a major risk as institutions move towards greater dependence on global, electronic transactions. The volume of transactions is increasing, but of even greater concern (and opportunity), the number of access points into the data network is growing rapidly.

It is interesting to note that today most banks compute models of their own customer buying behaviors to detect potential misuse of customer cards and accounts. It is these types of models we seek to compute over large populations of credit card users and hence our first prototype system focuses on credit card transaction data.

Fraud detection and prevention is strategic to the banking community. The JAM project has these objectives:

- Reduce the cost of fraud through **timely detection** and ongoing preventive measures.
- **Minimize the losses** by catching fraud more rapidly and minimize costs associated with false alarms.
- **Far exceed the level of protection by coordination** rather than what might be achievable through a fragmented, individual-institution approach to fraud detection.

Our work addresses the two primary technical components of a system to attack electronic fraud and meet these objectives:

- **Local fraud detection agents** to learn how to detect fraud and provide fraud detection services within a single corporate institution.
- **A secure, integrated meta-detection system** that joins the knowledge acquired by individual local agents to gain a global view of the network transactions and uses this vantage point to better detect fraud within institutions and enable the detection of dispersed attacks.

This system will rely on a secured and trusted network agent architecture. In this architecture, fraud detection agents can migrate their learning upward, and the meta-detector can provide its services downward to individual organizations. The system honors the need for privacy and data boundaries while still capturing most of the value of a global view of electronic transaction data.

The local, intra-institution fraud detection modules extract patterns of fraud from within an institution, or between it and another institution. Such detectors are specialized to the data schema of the database at the home institution.

These localized detectors are the raw material of a meta-learning process under which the locally learned fraud patterns are generalized. The meta-learning occurs when cooperating agents (fraud detection modules) share high level fraud descriptions (without compromising privacy or competition) and incorporate this into their local models. In addition, a global detector is constructed by meta-learning the local detectors to out-perform any of the individual detection agents.

If this concept is successful, the resultant meta-fraud-detector will be able to pick up patterns of fraud that are not detectable at the local level while at the same time, propagating the information about locally detected fraud patterns so that other financial institutions can be protected.

## 2.2 The Detection Components

The field of machine learning has made substantial progress, both empirically and theoretically, and a number of algorithms have been popularized and applied to a host of applications in diverse fields with very good success. Thus, we may simply apply the current generation of learning algorithms to an integrated collection of very large databases and wait for a classifier to be computed. However, the question is how long might we wait to compute an accurate classifier over widely distributed and very large databases? Many organizations seeking added value from their data are already dealing with overwhelming amounts of global information that in time will likely grow in size faster than available improvements in machine resources. Indeed, do the current generation of machine learning algorithms **scale** from tasks common today that include thousands of data items to new learning tasks encompassing perhaps as much as two orders of magnitude or more of data? The problem of course is further exacerbated by physical distribution of data that cannot possibly be localized at any one processing site for a host of practical or legal reasons. In such situations, it is infeasible to inspect all of the data at one processing site to compute one primary “global” classifier. We call the problem of learning useful new knowledge from large inherently distributed databases the *scaling problem for machine learning*.

Our approach to solve the scaling problem is to execute a number of learning processes (each implemented as a distinct serial program) on a number of data subsets (a *data reduction* technique) in parallel (eg. over a network of separate processing sites) and then to integrate and combine the collective results through a process we call *meta-learning*. Here, meta-learning serves as the means of “gluing” multiple knowledge sources together.

Meta-learning [1] is defined as learning of meta-knowledge about learned knowledge. In our work we concentrate on learning from the output of concept learning systems. In this case meta-learning means learning from the *predictions* of a set of classifiers on common *training data*. Thus, we are interested in the output of the classifiers, not the internal structure and strategies of the learning algorithms themselves. (Moreover, in several of the schemes we define, the training data presented to the learning algorithms initially are also available to the meta-learner under certain circumstances.) We have proposed several schemes for meta-learning. Here

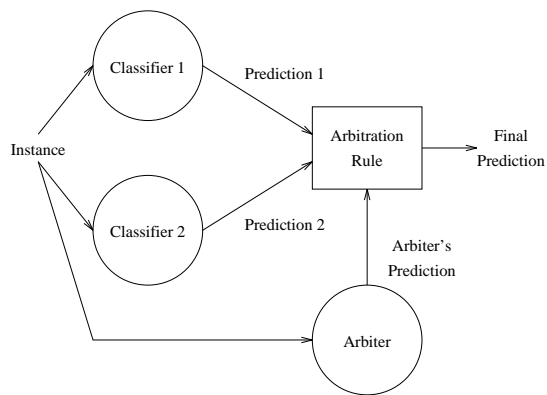


Figure 1: An arbiter with two classifiers.

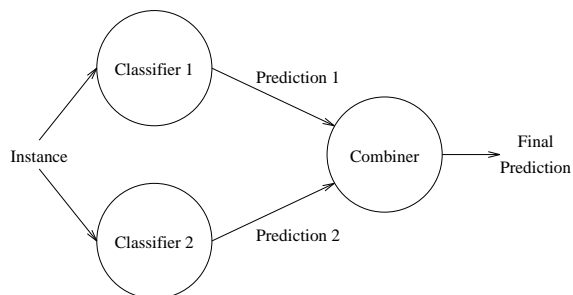


Figure 2: A combiner with two classifiers.

we briefly describe two strategies, the *arbiter* and *combiner*.

An *arbiter* [5] is learned by some learning algorithm to arbitrate among predictions generated by different base classifiers. This arbiter, together with an *arbitration rule*, decides a final classification outcome based upon the base predictions. Figure 1 depicts how the final prediction is made with predictions input from two base classifiers and a single arbiter.

In the *combiner* [2] strategy, the predictions of the learned base classifiers on the training set form the basis of the meta-learner’s training set. A *composition rule*, which varies in different schemes, determines the content of training examples for the meta-learner. From these examples, the meta-learner generates a meta-classifier, that we call a *combiner*. In classifying an instance, the base classifiers first generate their predictions. Based on the same composition rule, a new instance is generated from the predictions, which is then classified by the combiner. (see Figure 2). The aim of this strategy is to “correlate” the predictions from the base classifiers by learning the relationship between these predictions and the correct prediction. A combiner computes a prediction that may be entirely different from any proposed by a base classifier, whereas an arbiter chooses one of the predictions from the base classifiers and the arbiter itself.

Here we elaborate one strategy, the *class-combiner*. The *class-combiner* strategy seeks to learn the correlations of predictions among a number of base classifiers. For this purpose, a separate data set, called the meta-learning validation set, is applied to every base classifier. The predictions by the base classifiers along with the true label of the input data item reveals the correlation of predictions among

Class	Attribute vector	Example	Base classifiers' predictions		
$class(x)$	$attrvec(x)$	$x$	$C_1(x)$	$C_2(x)$	$C_3(x)$
table	$attrvec_1$	$x_1$	table	table	table
chair	$attrvec_2$	$x_2$	table	chair	lamp
lamp	$attrvec_3$	$x_3$	lamp	chair	chair

Training set for the <i>class-combiner</i> scheme	
Class	Attribute vector
table	(table, table, table)
chair	(table, chair, lamp)
lamp	(lamp, chair, chair)

Figure 3: Sample training sets generated by the *class-combiner* strategy

the base classifiers. The base classifier predictions as well as the correct class label are used to train the meta-level classifier. A particular example is displayed in Figure 3.

In prior publications, a variety of inductive learning algorithms have been employed to generate the appropriate classifiers and meta-classifiers and their performance has been rigorously measured and reported. The following is a brief summary of meta-learning results that are pertinent to the fraud detection environment. A detailed result and discussion can be found in [1].

- The meta-learning strategies do show a consistent improvement in classification accuracy over any of the base classifiers trained on a subsets of available training data. Our studies show that classifiers trained individually from random subsets of a large data set are not as accurate as combining a collection of separately learned classifiers. Thus, applying a learning algorithm to “random” data available at only one distributed site may not produce a classifier that is as informed or knowledgeable than a meta-learned classifier computed over two or more sites with more available data.
- Under the arbiter tree strategy allowing unbounded meta-level training sets, we determined that, over the variety of algorithms and learning tasks employed, at most 30%, and in certain cases at most 10%, of the entire training data was required at any one processing site to maintain the equivalent predictive accuracy of a single global classifier computed from one learning algorithm applied to all available data. This provides evidence that distributing the learning task is indeed possible with limited local resources. Thus, if the total amount of data is so large as to be impossible to process at any one site, the arbiter tree meta-learning strategy may provide an alternative distributed processing approach that exhibits the ability to learn fully what is available only partially at any one site.
- The combiner tree strategy was demonstrated to consistently boost the predictive accuracy of a global classifier under certain circumstances. This suggests that a properly configured meta-learning strategy combining multiple knowledge sources provides a more accurate view of all available data than any one

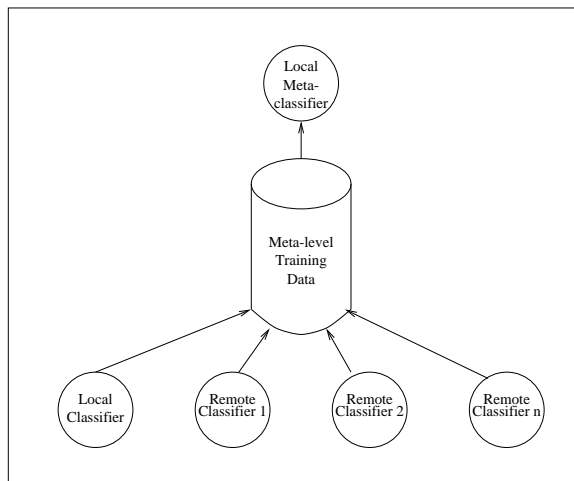


Figure 4: Sharing Knowledge Without Sharing Data

learning algorithm alone can achieve. Also, training combiner trees is more efficient than training arbiter trees.

These results lead us to conclude that a larger scale test of meta-learning architectures is warranted. For this reason, we have teamed with the FSTC to apply these concepts to the task of learning about fraudulent transactions for use in intrusion detection in large networked financial information systems. A critical issue is whether the proposed approach exhibits good predictive accuracy in practice. For this reason we are building JAM and applying it to credit card fraud transaction data. Our initial results are described in a later section. We first describe how such a system might be deployed in practice.

### 2.3 How Local Detection Components Work

Consider the generic problem of detecting fraudulent transactions in which we are not concerned with global coordinated attacks. We posit a good candidate approach for architecting a distributed meta-learning system as follows. Here  $DB$  refers to (archived) transaction data known to be either fraud or non-fraud held by each of the participating banks in a networked financial information system.

1. Each bank  $i, 1 < i \leq N$ , uses some learning algorithm or other on one or more of their databases,  $DB_i$ , to produce a classifier  $f_i$ . In the simplest version, all the  $f_i$  have the same input feature space. Note that each  $f_i$  is just a mapping from the features space of transaction,  $x$ , to a bimodal fraud label.
2. Every bank  $i$  sends to every other bank its  $f_i$ . So at the end of this stage, each bank has a copy of all  $N$  classifiers,  $f_1, \dots, f_N$ .
3. Each bank  $i$  had held separate some data, call it  $T_i$ , from the  $DB_i$ . Each bank then uses  $T_i$  and the set of all the  $f$ 's to learn the mapping from  $\langle f_1(x), f_2(x), \dots, f_N(x), x \rangle$  to a fraud label. A separate classifier  $F_i$  is trained on this set of mappings.

4. Each bank uses its  $F_i$  use as a data filter to mark and label incoming transactions with a fraud label.

This approach is depicted in figure 4. There is an issue of how  $T_i$  is formed. One approach is to simply use a “hold out” set from  $DB$  that is not used to train classifiers. Another approach employed here in our experiments is as follows:

- Split the training data  $DB$  at each site into 2 halves:  $DB_1$  and  $DB_2$ .
- Train a classifier  $f_1$  on  $DB_1$  and apply  $f_1$  together with all other remote classifiers on  $DB_2$  to generate half of the meta-level training data.
- Repeat this on the other half of  $DB$  and generate another half of the meta-level training data.
- re-train the local classifier  $f$  on the entire local partition,  $DB$ .

Our results detailed in section 4 shows that this approach works well. An important feature here is that each bank only exchanges its locally computed classifiers, and not their local data. Thus, we are able to demonstrate the principle that it is possible to “share knowledge without disclosing data”.

### 3 The JAM Architecture

JAM [8] is architected as an agent based system, a distributed computing construct that is designed as an extension of OS environments. It is a distributed meta-learning system that supports the launching of learning and meta-learning agents to distributed database sites. JAM is implemented as a collection of distributed learning and classification programs linked together through a network of *Datasites*. Each JAM Datasite consists of:

- A local database(s),
- One or more learning agents (words machine learning programs) that may migrate to other sites as JAVA applets, or be locally stored as native applications callable by JAVA applets,
- One or more meta-learning agents,
- A local user configuration file,
- Graphical User Interface and Animation facilities.

The JAM Datasites have been designed to collaborate with each other to exchange classifier agents that are computed by the learning agents. First, *local learning agents* operate on the *local database* and compute the Datasite’s local classifiers. Each Datasite may then import (remote) classifiers from its peer Datasites and combine these with its own local classifier using the *local meta-learning agent*. Finally, once the base and meta-classifiers are computed, the JAM system manages the execution of these modules to classify and label datasets of interest.

JAM’s source code can be downloaded for research purposes at the URL: <http://www.cs.columbia.edu/~sal/JAM/PROJECT>.

## 4 Performance

We ran a series of experiments using JAM and its component learning programs to validate the effectiveness of meta-learning over distributed credit card fraud data. First, we report the measured performance of the learning algorithms and the derived meta-classifiers for a randomly chosen sample of the full data set. We then report the measured performance of the derived meta-classifiers and the constituent base classifiers computed by JAM on the entire set of credit card data distributed among 6 processing sites.

### 4.1 Credit Card Fraud Transaction Data

We have obtained a large database with 500,000 records, of credit card transaction data from one of the member banks of the FSTC. Under the terms of the non-disclosure agreement, we can not reveal the details of the schema beyond the following general description about the data.

- Number of Attributes:  $30 + -\Delta$  (all numeric)
  - Many fields are categorical (i.e. numbers represent a few discrete categories)
  - Developed over years to capture important information
- Size:
  - Record length: Fixed 137 bytes per transaction
  - Number of records: 500,000
  - Sample distribution: 20% fraud/ 80% Non-fraud drawn 42,000 per month
  - Time Frame: Span 11/95 - 10/96
- Type of Information:
  - A (jumbled) account number (no real identifiers)
  - Scores produced by a COTS authorization/detection system
  - Date/Time of transaction
  - Past payment information of the transactor
  - Amount of transaction
  - Geographic information: where the transaction was initiated, the location of the merchant and transactor
  - Codes for validity and manner of entry of the transaction
  - An industry standard code for the type of merchant
  - A code for other recent “non-monetary” transaction types by transactor
  - The age of the account and the card
  - Other card/account information
  - Confidential/Proprietary Fields (other potential indicators)
  - Fraud Label (0/1)

Our task is to compute a classifier that predicts whether a transaction is fraud or non-fraud from this database. As for many other data mining tasks, we need to consider the following issues in our experimental set up.

- **Data Conditioning:**
  - First, in our experiments, we removed several redundant fields from the original data schema. This helped to reduce the database size, thus speeding up the learning programs. We have compared the results of learning on the conditioned data versus the original data, and saw no loss in accuracy. We also discretized some temporal fields and found them to be of no predictive value.
  - Second, since the data has a skewed class distribution (20% fraud and 80% non-fraud), can we train on data that has (artificially) higher fraud rate and compute more accurate fraud patterns? And what is the optimal fraud rate in the training data?
- **Data Sampling:** Most of the machine learning algorithms require the entire training data be loaded into the main memory. Since our database is very large, this is not practical. In our initial experiments, only a portion of the original database was used for learning (details provided in the next section). When the full dataset was used, partitions of the data were distributed among separate processing sites.
- **Testing:** In our experiment, the training data were sampled from earlier months, the testing data were sampled from later months. The intuition behind this scheme is that we need to simulate the real world environment where models will be learned using data from the previous months, and used to classify data of the current month.
- **Accuracy Evaluation:** Overall classifier accuracy as measured against the test set is an important metric in many cases. However, in situations as these where the sample is highly skewed (20% of the test data has the target label), we can design the simplest of classifiers that always "guesses" the majority label. It will therefore be 80% accurate! In this case, the *True Positive (TP) Rate* (i.e. the *Fraud Catching Rate*) as well as the *False Positive (FP) Rate* (i.e. the *False Alarm Rate*) are the critical measures. A low fraud catching rate means that many fraudulent transactions will go undetected by our system. On the other hand, a high false alarm rate will block many legitimate transactions at great expense.

## 4.2 Performance of Machine Learning and Meta-learning on Credit Card Data

One set of experiments was designed to reveal how machine learning and meta-learning behaves in this domain. We aim to find out the relative performance of meta-learning versus base level learning and the best combinations of base-learning algorithms to use to generate the best (meta-)classifiers. Other problems we addressed include finding the best distribution to use in training sets as well as metrics to select base classifiers to form the best meta-classifiers. The detailed discussion about these last two issues can be found in [9].

Bearing these particular goals in mind, we have formed the following learning task.

- Data of a full year are partitioned according to months 1 to 12. Data from months 1 to 10 are used for training, data in month 11 is used to generate meta-level training data and data in month 12 is used for testing. This simulates the real world situation of fraud detection where data in the past are used to learn patterns and these patterns are used to predict fraud on new data from the present.
- The same percentage of data is randomly chosen from months 1 to 10 for a total of 42000 records for training.
- A randomly chosen 4000 data items from month 11 are used for meta-learning validation and a random sample of 4000 records from month 12 is chosen for testing against all learning algorithms (including base-level learning and meta-learning).
- The 42000 record training data is partitioned into 5 parts of equal size:
  - $f$  pure fraud
  - $nf_1, nf_2, nf_3, nf_4$  pure non-fraud
  - Each  $nf_i$  is partitioned into 2 sub-parts of equal size and forms into  $nf f_1$  to  $nf f_8$ .
- Recall that the 20% frauds in training data is a relatively skewed distribution. It has been reported informally that forming training distributions that are more balanced will lead to classifiers that have higher accuracy. The following distribution and partitions are therefore formed:
  - 67%/37% fraud/non-frauds in training data: 4 partitions by concatenating  $f$  with  $nf_1, nf_3, nf_5, nf_7$ .
  - 50%/50% fraud/non-fraud in training data: 4 partitions by concatenating  $f$  with each  $nf_i$ .
  - 33%/67% fraud/non-fraud in training data: 3 partitions by concatenating  $f$  with 2 randomly chosen  $nf_i$
  - 25%/75% fraud/non-fraud in training data: 2 partitions by concatenating  $f$  with 3 randomly chosen  $nf_i$
  - 20%/80% the original 42000 records sample data set
- Four learning algorithms were available and used: ID3 [11], CART [12], BAYES [13] and RIPPER. Each is applied to every partition above. ID3 and CART is part of IND package [17]. BAYES is a naive Bayesian classifier. RIPPER [10] is a rule learning program obtained from William Cohen at ATT.
- In this experiment, we have chosen to use the *class-combiner* meta-learning technique. The results in this paper detail the performance of meta-classifiers that combine base classifiers with the highest TP rate. (Other strategies are possible and under investigation.)

The results displayed are computed averages of 3-fold cross validation. We display the six best (meta-)classifiers. (A more detailed description of the full set of results can be found in [9] and at our project home page.)

Classifier Name	True Positive Rate	False Positive Rate
BAYES as meta-learner combining 4 base classifiers (learned over 50%/50% distribution) with highest True Positive Rate	80%	13%
RIPPER trained over 50%/50% distribution	80%	16%
CART trained over 50%/50% distribution	80%	16%
BAYES as meta-learner combining 3 base classifiers (learned over 50%/50% distribution) with least correlated error [16]	80%	17%
BAYES as meta-learner combining 4 classifiers learned over 50%/50% distribution with least correlated error rate	76%	13%
ID3 trained over 50%/50% distribution	74%	23%

Table 1: Results of Best Classifiers

In Table 1, we show results of the few best performers. The best classifiers have a TP Rate of around 80% and an FP Rate of below 20%. The best classifier overall is a meta-classifier: BAYES used as the meta-learner to combine 4 base classifiers with the highest TP Rate (each trained on a 50%/50% fraud/non-fraud distribution). The next 2 best classifiers are base classifiers: CART and RIPPER each trained on a 50%/50% fraud/non-fraud distribution. The TP Rate of 3 base classifiers are identical, but the meta-classifier has the lowest FP Rate. Other findings (detailed in [9]) include that 50%/50% fraud distribution in training data will generate classifiers with highest TP rate and relatively low FP rate.

This set of experiments provides some information about the relative performance of several inductive learning algorithms and meta-learning strategies. Armed with this, we have carried out another set of experiments to evaluate the JAM’s ability to compute classifiers over remote processing sites.

### 4.3 Performance of JAM on Credit Card Fraud Data

In a separate experiment, we analyzed the complete 500,000 data items on JAM using 6 sites. This experiment simulates the scenario where six banks compute classifiers over their privately held data, exchange classifiers with each other, and then learn a local meta-classifier. (The meta-learning strategy employed here is *class-combiner*.)

The experimental set up is as follows:

- The complete data (some 500,000 items) are partitioned into 12 parts,  $m_1$  to  $m_{12}$ , according to month. The ‘date’ field in the data is removed to avoid bias.
- We have 6 sites running JAM. Each site represents a different bank.
- Each site has 2 months worth of data. One data set is used to produce the local classifier, the other one is treated as the new data for accuracy evaluation.
- During training, some random selection of non-frauds of the first month data were removed to produce a 50%/50% fraud/non-fraud training distribution.
- The 2nd month data used for testing maintains 20%/80% original fraud distribution.
- Two of JAM’s processing sites run CART as the base learning algorithm, while four others run ID3. This choice was purely arbitrary. CART, BAYES and RIPPER are used at all 6 sites as the meta-level learner.
- The meta-level classifier was trained from the data set generated by the approach introduced in Section 2.3. Each half of the local data at each site is used to generate a classifier that generates half of the meta-level training data.

The detailed results are shown in Table 2. It is obvious that meta-learning’s performance is consistently better than any of the base classifiers through out the 6 sites that are measured. Notice that the best meta-classifier has a TP rate as high as 8% better than the next best classifier, and has either slightly lower or equivalent FP rate than any of the 6 base classifiers. This boost in accuracy is seen throughout all the 6 sites under study. To display the positive contribution of TP and negative contribution of a FP, we simply compute and display the difference between these rates to compare the different classifiers. At all the 6 sites tested, the 3 meta-classifiers are the best classifiers at each site. These results show that sharing knowledge without sharing data is not only feasible, but also is very useful!

#### 4.4 Cost Model Performance on Credit Card Fraud Data

The experiments shown do not incorporate any appropriate cost model. In this and other domains, simply measuring TP and FP rates does not tell the whole story. Each transaction has an associated cost, the transaction amount *tranamt*, and hence we seek methods that produce classifiers that generate minimal loss in dollars. The ideal situation would be to compute classifiers with a 100% TP rate, meaning all fraudulent transactions would be caught, no dollar losses would be incurred. Unfortunately, this is not possible. Each transaction predicted to be fraudulent requires an “overhead” referral fee for authorization personnel to decide a final disposition. This “overhead” cost is typically a *fixed fee* that we call \$X. Therefore, even if we can accurately predict and identify all fraudulent transactions, those whose transaction amount is less than \$X would produce \$X in losses anyway.

In a real world context, this immediately implies that any transaction whose amount is less than \$X would simply be untested and authorized. Hence, for those transactions we only lose the transaction amount of the authorized frauds!

We apply the following aggregate cost model (dollars lost) to evaluate the performance of different classifiers in the following experiments. We seek to minimize these costs and display our results in terms of average loss per transaction.

- $Cost(FN) = tranamt$ , here we loose the transaction amount for those fraudulent transactions mislabeled legitimate;
- $Cost(FP) = X$  if  $tranamt > X$  or 0 if  $tranamt \leq X$ . Transactions less than \$X are ignored, and hence we suffer no referral fee penalty. Legitimate transactions greater than \$X incorrectly labelled fraudulent incurs the fixed referral fee as a loss.
- $Cost(TP) = X$  if  $tranamt > X$  or  $tranamt$  if  $tranamt \leq X$ . Note that even for correctly predicted fraudulent transactions whose amount is greater than \$X, we still loose the fixed referral fee.
- $Cost(TN) = 0$ . Unchallenged legitimate transactions incur no cost.

In this more realistic context, we seek methods to identify fraudulent transactions whose amounts are greater than \$X resulting in minimal dollar losses. We ran a set of experiments using different learning algorithms (RIPPER, CART, C4.5, BAYES and Meta-learning (by *class-combiner* approach) and computed the aggregate dollar losses for each. The *class-combiner* meta-learning strategy applied in this experiment partitions the training data in 2 equal halves. We train base level classifiers on one half and validate these using the other half to generate half of the meta-level training data. This process is repeated by reversing the roles of the two halves of training data forming a final meta-level training set. Each base learning algorithm is applied to the entire training data to generate base classifiers which are then combined by the meta-classifier trained on the meta-level training data.

In this task, we have used training data from October 1995 to July 1996 for a total of 10 months. The complete August 1996 data are used for testing. We fix the training data set size to be 6,400 (since 6,400 is the number that produces 90% fraudulent transaction in the training data with no replicates). We formed training sets from the above 10 months varying the training distributions from 10% to 90% frauds.

In Figure 5, we show the aggregate cost of RIPPER, CART, C4.5, BAYES and Meta-learning with the *class-combiner* strategy. In each figure, we display 6 curves. Each one shows the change in the “average per transaction loss” while varying the fixed referral fee. We also display the relative costs while varying the distribution of fraudulent transactions in the training data. Each point on a curve is the average of 10 classifiers trained on training data from 10 different months.

The trends displayed in the curves by all learners, except for BAYES, are very similar. For any referral fee \$X, the average dollar loss per transaction reduces when the training data fraud rate goes from 10% up to 50% or so and begins to pick up afterwards. The BAYES curve is quite flat. For RIPPER, CART and C4.5, the smallest dollar losses appear at the level where fraudulent transactions are about equal with legitimate transactions in the training sets. For Meta-learning, the least cost appears between 50% and 70%. Comparing the result of RIPPER, CART, C4.5, they are very close in their cost results. Naive BAYES is significantly worse than any other method.

One interesting finding is that the *class-combiner* strategy incurs the least dollar losses for any of the base classifiers. This is nearly entirely consistent over any referral fee and training data distribution. The lowest cost at each curve for the *class combiner* meta-learner (each with a different referral fee) is lower than corresponding results by RIPPER, CART and C4.5.

## 5 Discussion and Future Work

The best TP Rate displayed in these initial results is approximately 80%, with a relatively high FP Rate of about 25%. This means that 25% of legitimate transactions will produce false alarms. In other work, we have devised better meta-learning techniques that display remarkably reduced FP rates. One such method is called *conflict-resolver*. The idea is to analyze the meta-level training data generated by the base classifiers to identify conflicting meta-level training data. In a separate report, we detail this strategy that has produced meta-classifiers with 87% TP and 9% FP. (Browse our project web site for details.)

The data we ran on JAM was provided by one bank collaborator. Recently we acquired another data set from another bank. The two schemas involved do not exactly match even though nearly all the fields are common among both banks. Hence we are faced with a version of the “schema integration problem” to deal with those fields that do not exactly match each other. This naturally complicates the matter of how to appropriately integrate base classifiers trained over slightly different data sources. We are exploring several strategies and will report upon the results when they are available.

## 6 Conclusion

We have shown meta-learning provides an important step in developing systems that learn from massive databases, and that scale. A deployed and secured meta-learning system will provide the means of using large numbers of low-cost networked computers who collectively learn from massive databases useful and new knowledge, that would otherwise be prohibitively expensive to achieve. We have also shown promising results by running JAM on credit card fraud data demonstrating the effectiveness of the approach for fraud detection and network security. In the results reported, we have shown that 50% fraud distribution in training data generate classifiers with both high TP and low FP rates and the lowest per transaction loss due to fraud. We have also demonstrated that meta-learning methods consistently produce classifiers with higher accuracy and lower per transaction losses than any of the base level learners used. We believe meta-learning systems deployed as intelligent agents will be an important contributing technology to deploy fraud and intrusion detection facilities in global-scale, integrated information systems.

## Acknowledgments

We wish to thank David Wolpert, formerly of TXN and presently at IBM Almaden, Hank Vacarro of TXN, Shaula Yemini of Smarts, Inc. and Yechiam Yemini for

many useful and insightful discussions. Their early collaboration with us in this effort helped sharpen many of the concepts described in this paper. They also contributed to the exposition of the concepts in this paper by reviewing earlier drafts. We also wish to thank Dan Schutzer of Citicorp, Adam Banckenroth of Chemical Bank and John Doggett of Bank of Boston, all executive members of the FSTC, for their support of this work. This work is supported by the Intrusion Detection Program (BAA9603) from DARPA (F303602-96-1-0311), NSF (IRI-96-32225 and CDA-96-25374) and NYSSTF(423115-445).

## References

- [1] P. Chan and S. Stolfo. An Extensible Meta-learning Approach for Scalable and Accurate Inductive Learning. *Ph.D. Thesis, Department of Computer Science, Columbia University, New York, New York*, 1997.
- [2] P. Chan and S. Stolfo. Experiments on multistrategy learning by meta-learning. In *Proc. Second Intl. Conf. Info. Know. Manag.*, pages 314–323, 1993.
- [3] P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proc. Second Intl. Work. on Multistrategy Learning*, pages 150–165, 1993.
- [4] P. Chan and S. Stolfo. Toward multistrategy parallel and distributed learning in sequence analysis. In *Proc. First Intl. Conf. Intel. Sys. Mol. Biol.*, pages 65–73, 1993.
- [5] P. Chan and S. Stolfo. Toward parallel and distributed learning by meta-learning. In *Working Notes AAAI Work. Know. Disc. Databases*, pages 227–240, 1993.
- [6] P. Chan and S. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc. Twelfth Intl. Conf. Machine Learning*, pages 90–98, 1995.
- [7] P. Chan and S. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proc. Intl. Conf. Knowledge Discovery and Data Mining*, pages 39–44, 1995.
- [8] S. Stolfo, A. Prodromidis, S. Tselepis, W. Lee, D. Fan and P. Chan. JAM: Java Agents for Meta-learning over Distributed Databases. In *Prod. Third Intl. Conf. Knowledge Discovery and Data Mining*, 1997.
- [9] S. Stolfo, D. Fan, W. Lee, A. Prodromidis and P. Chan. Credit Card Fraud Detection Using Meta-learning: Issues and Initial Results. In *Working Notes AAAI-97*, 1997.
- [10] W. Cohen. Fast Effective Rule Induction. In *Twelveth Intl. Conf. on Machine Learning*, pages 115–123, 1995.
- [11] R. Quinlan. Induction of Decision Trees. In *Machine Learning*, 1:81-106.
- [12] L. Breiman, J. Friedman, R. Olshen and C. Stone. *Classification and Regression Tree*, Wadsworth, Belmont, CA, 1984.
- [13] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–285, 1989.

- [14] T. M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Dept. Comp. Sci., Rutgers Univ., 1980.
- [15] L. Xu, A. Krzyzak, and C. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. Sys. Man. Cyb.*, 22:418–435, 1992.
- [16] K. Ali and M. Pazzani. Error Reduction through Learning Multiple Descriptions. In *Machine Learning*, vol.24, pages 173–202.
- [17] W. Buntine and R. Caruana. Introduction to IND and Recursive Partitioning, NASA Ames Research Center.
- [18] R. Branchman, T. Khabaza, W. Kloesgen and et all. Mining Business Databases. In *CACM*, vol.11, November 1997, pages 41–48

Base Classifiers	TP	FP	TP - FP
ID3	69%	31%	38%
ID3	73%	33%	40%
CART	63%	29%	34%
ID3	73%	26%	47%
ID3	75%	26%	49%
CART	69%	40%	29%
Meta Classifiers	TP	FP	TP - FP
CART	78%	27%	51%
BAYES	79%	31%	48%
RIPPER	78%	28%	50%

Base Classifiers	TP	FP	TP - FP
ID3	79%	39%	40%
ID3	68%	25%	43%
ID3	72%	26%	46%
CART	76%	20%	56%
ID3	74%	20%	54%
CART	68%	25%	43%
Meta Classifiers	TP	FP	TP - FP
CART	79%	39%	40%
BAYES	80%	21%	59%
RIPPER	75%	24%	51%

Base Classifiers	TP	FP	TP - FP
ID3	80%	30%	50%
ID3	81%	30%	51%
CART	85%	32%	53%
ID3	80%	23%	57%
ID3	82%	41%	41%
CART	74%	28%	46%
Meta Classifiers	TP	FP	TP - FP
CART	88%	31%	57%
BAYES	86%	25%	55%
RIPPER	86%	27%	59%

Base Classifiers	TP	FP	TP - FP
ID3	71%	29%	42%
ID3	67%	26%	41%
ID3	73%	21%	52%
CART	72%	21%	51%
ID3	80%	44%	36%
CART	64%	26%	38%
Meta Classifiers	TP	FP	TP - FP
CART	77%	27%	50%
BAYES	77%	28%	49%
RIPPER	78%	24%	54%

Base Classifiers	TP	FP	TP - FP
ID3	80%	30%	50%
ID3	81%	30%	51%
CART	85%	32%	53%
ID3	80%	23%	57%
ID3	82%	41%	41%
CART	74%	28%	46%
Meta Classifiers	TP	FP	TP - FP
CART	88%	31%	57%
BAYES	86%	25%	55%
RIPPER	86%	27%	59%

Base Classifiers	TP	FP	TP - FP
ID3	71%	29%	42%
ID3	67%	26%	41%
ID3	73%	21%	52%
CART	72%	21%	51%
ID3	80%	44%	36%
CART	64%	26%	38%
Meta Classifiers	TP	FP	TP - FP
CART	77%	27%	50%
BAYES	77%	28%	49%
RIPPER	78%	24%	54%

Table 2: Result of running JAM on 6 sites

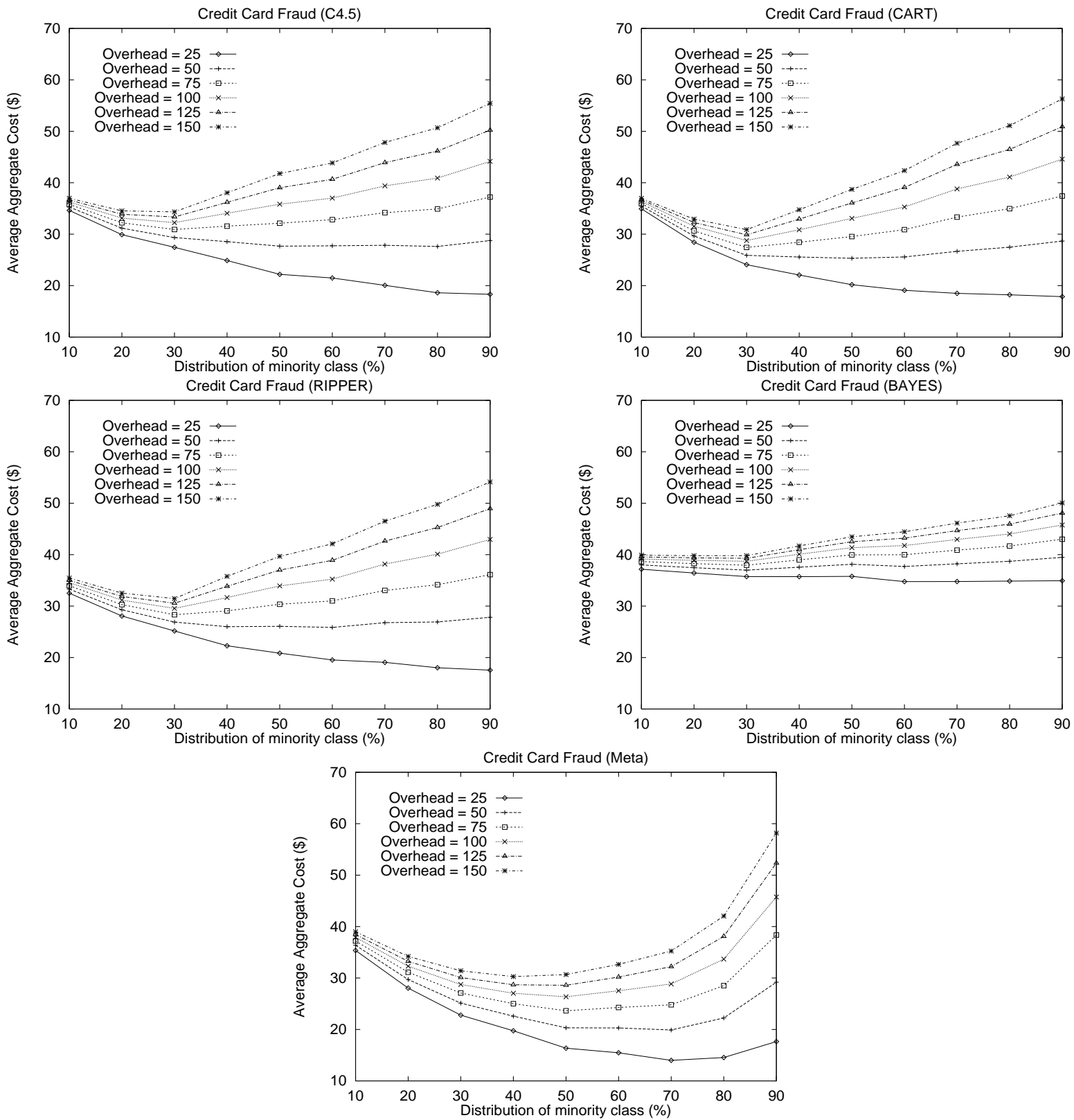


Figure 5: Aggregate Cost Result